SOFTWARE CHANGE MANAGEMENT: BALANCING FLEXIBILITY AND CONTROL

Barbara Pfarr

NASA/GSFC
GSFC/584.0/Greenbelt, MD/USA
Fax:301-286-1602, E-mail: barbara.pfarr@gsfc.nasa.gov

Bradley Boyce

NASA/GSFC/Orbital Sciences Corporation
Vision 2000 Co-location Facility, Lanham MD
Fax:301-918-7474, E-mail:bboyce@v2pop.hst.nasa.gov

Darlene West

NASA/GSFC/Orbital Sciences Corporation
Vision 2000 Co-location Facility, Lanham MD
Fax:301-918-7474, E-mail:dwest@v2pop.hst.nasa.gov

ABSTRACT

The Hubble Space Telescope (HST) is an extremely valuable scientific resource, and any changes to the ground system software must be carefully controlled to avoid harming the telescope. Traditionally, space operations software that creates and sends commands to a spacecraft and receives telemetry from a spacecraft is carefully configured, and changes to the baseline system require lengthy manual review and approval processes.

It is desirable to have processes for modifying operational ground system software that minimize the change delay without compromising spacecraft safety. If the processes can be automated and made available to all interested users, the waiting time can be significantly reduced over a manually intensive processes. By creating and using new state-of-the art change management tools and modifying the change management process, we reduce the time and effort required to make software and database changes. These improvements will reduce user frustration, improve user performance, and reduce overall maintenance costs.

This paper discusses how the Maintain and Upgrade the Ground System (MUGSy) was designed and implemented to minimize the time to identify, implement, review and approve changes to the HST ground system software and shared databases.

1. THE NEED FOR RE-ENGINEERING THE SOFTWARE CHANGE MANAGEMENT SYSTEM

The major goal of the Hubble Space Telescope (HST) Control Center System (CCS) re-engineering project, known as Vision 2000, was to reduce the cost of HST operations, including system development and maintenance by at least 50 percent by the year 2000. Additional goals included introducing automation, where appropriate, to eliminate routine, repetitive operational procedures and functions, and to use Commercial-off-the-shelf products (COTS) wherever cost effective. A specific sub-goal stated in the 1995 Vision 2000 operations concept was that configuration management should add zero wait time to the change process. A further challenge was to integrate the change management system into the overall ground system. The goal of reducing staffing levels provided an additional incentive to simultaneously re-engineering the change process and the spacecraft control and analysis processes.

The Change Management System in place in 1995 was given a critical examination. The system at that time was totally manual and highly people intensive. Computer-editable forms were only beginning to replace paper forms, but there were no on-line systems to manage the information. Each sub-project within the HST project had a configuration management team to manage the forms and configuration control boards that were responsible for reviewing and approving all changes. The entire change management process was independent of the software development systems being used. The examination of the existing (1995) process yielded an 8 page business process diagram that included 54 steps, 10 roles, and 23 hand-offs. The cycle time for a desired change ranged from 4 weeks to 12 months. Minor changes took somewhat less time than major, cross-organizational changes, but still were slowed down by a process designed to handle all changes in essentially the same way.

2. DEFINING THE NEW MAINTAIN AND UPGRADE THE GROUND SYSTEM (MUGSY) PROCESS

The new process, called “Maintain and Upgrade the Ground System (MUGSy)” was developed from studying the exhaustive diagram of the existing system and modifying it to address the Vision 2000 goals, one of which was to implement a change management system that had the flexibility to respond to different levels of problems in different ways. Changes completely within a specific subsystem should be made and controlled by the local user/maintainer team associated with that subsystem. Changes affecting multiple subsystems should be controlled depending on applicability, by either cross-organizational, functionally oriented working groups, a project-wide system engineering board or upper management.

A similar process for “Maintaining the Office Environment” was merged with MUGSy. Combining these two processes was possible because future office tools are expected to be compatible with future development/maintenance tools (i.e., workstations with integrated tool suites). Steps that involved duplication or re-work were redesigned or eliminated. The process was streamlined until only value-added steps remained. Then, systems to automate the processes, described later in this paper, were identified wherever possible to increase productivity and reduce staffing levels. Change Control Boards (CCB) were eliminated for sub-system changes. The resulting process consisted of only 19 steps and involved 5 roles and 5 handoffs. The resulting cycle times are now from 1 day to 12 weeks, a significant reduction from the 1995 cycle times.

3. SPECIFIC REQUIREMENTS FOR MUGSY

The goals of Vision 2000 led to the following specific requirements for the implementation of the MUGSy process. Users must be able to submit Change Requests (CRs) from any platform (PC, MAC, UNIX) and be able to access CR status from any platform. Owners of products need to be notified immediately of submitted CRs. Users must follow a pre-defined process that allows for flexibility in handling different levels of change. Report generation should be automatic. The system should provide a method for capturing solutions or comments. The automated portion of the process should be built using COTS products, if available, and should minimize the need for manual work. Finally, the initial version of the system had to be in place in time for release 1.0 of the CCS effort to capture the CRs from the beginning. It needed to permit users to request changes to MUGSy itself, to grow and change as the usage of the system increased.

The key differences between the new environment and the old are as follows: development and testing will not be done in a separate environment; the originator of a request, whenever possible, is empowered to make the change or test it; documentation is all electronic; most management approval steps have been eliminated; and separate systems for discrepancies versus changes have been eliminated. A key enabler of the new system was to create a single unified development group under one manager.

4. EVALUATION OF COTS PRODUCTS

The evaluation of potential COTS products for MUGSy implementation was conducted in four phases: 1. The MUGSy requirements were developed; 2. A literature search was conducted; 3. Demonstrations of the "best-fit COTS products" were conducted; and, finally 4. The product was selected. The total evaluation period took six months. In some cases, these phases overlapped.

5. MUGSy IMPLEMENTATION SCHEDULE

The remainder of this paper will specifically address the MUGSy Change Request Management (CRM) implementation. The MUGSy implementation schedule was as follows: COTS products were selected in April 1996 and ordered immediately. A working system was required to support CCS development by October 1, 1996. The vendors delivered the products quickly and provided training in May 1996. Release 1 was delivered on September 30 and was fully operational to support the integration and test of the CCS Release 1. This was the first of 16 deliveries over the next 2 years. Currently, MUGSy version 3.1c is operational.

6. MUGSy DESIGN AND IMPLEMENTATION

The MUGSy Design and Implementation was a very different paradigm than traditional system design and implementation. Because the selected CRM COTS product provided a rich "out-of-the-box" capability, the design consisted primarily of identifying areas where customization was required. The team's work was composed of three main tasks: 1. define the on-line screens/forms; 2. define the process and business rules for notification; 3. define customization for Internet interface presentation.

The design of the on-line screens and forms required an assessment of the COTS product pre-defined database schema. The key task was to determine how to maximize schema usage and to identify the customization required. Any changes to the pre-defined schema required subsequent changes to the "pre-defined forms and screens". In addition, the pre-defined tables needed to be populated and mandatory fields needed to be defined. Three table sets were identified as required: Change Requests, Products, and Contacts. Each of these table sets had three operations: SUBMIT, UPDATE, and VIEW. The VIEW operation was available to everyone while SUBMIT and UPDATE were only available to "privileged users". The three table sets along with a state table and business rules interact to form the CR process. The Change Request Table stores the actual CRs, written against products. The Product Table provides users with a predefined set of products to write CRs against. The Contacts Table defines the owner and approver for each product. Typical online access is through the MUGSy Find/Update Change Request Screen, as depicted in Figure 1.

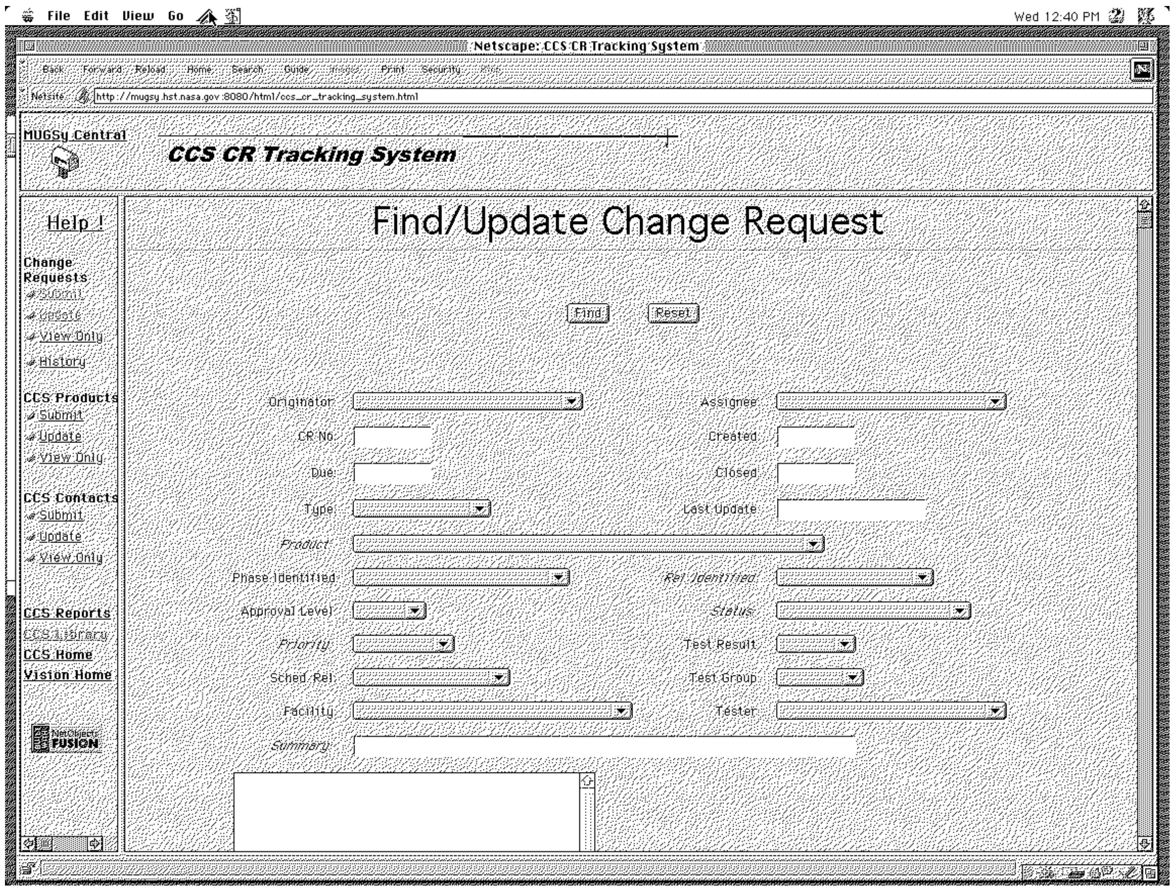

Figure 1 MUGSy Find/Update CR Form

The process and business rule design was based on the new MUGSy business "process". The process design populated the COTS state table specifying the allowable states and state transitions. This effectively defined and enforced the process. The original implementation consisted of 9 states: 1. Reported, 2. Assigned, 3. Analysis Complete , 4. Solution Approved, 5. Upgrade Complete, 6. Test Complete, 7. Deployment Approved, 8. Deployed, 9. Closed. The states were used to pre-populate the CR "status". To streamline the process in later releases, Analysis Complete, Solution Approved, Deployment Approved and Deployed were made optional. The business rules defined which roles (defined in the Contacts Table set) were notified electronically for each state change (defined in the state table). The Process and Business Rules Implemented in MUGSy are summarized in Figure 2. The required process states are indicated by the gray colored boxes.

The final MUGSy design step defined the customization required to present the Internet interface. This required a skilled Hyper Text Markup Language (HTML) developer to "create an appealing wrapper" around the COTS produced HTML and Common Gateway Interface (CGI). Graphics and hyperlinks to the COTS-generated CR forms and screens were added.

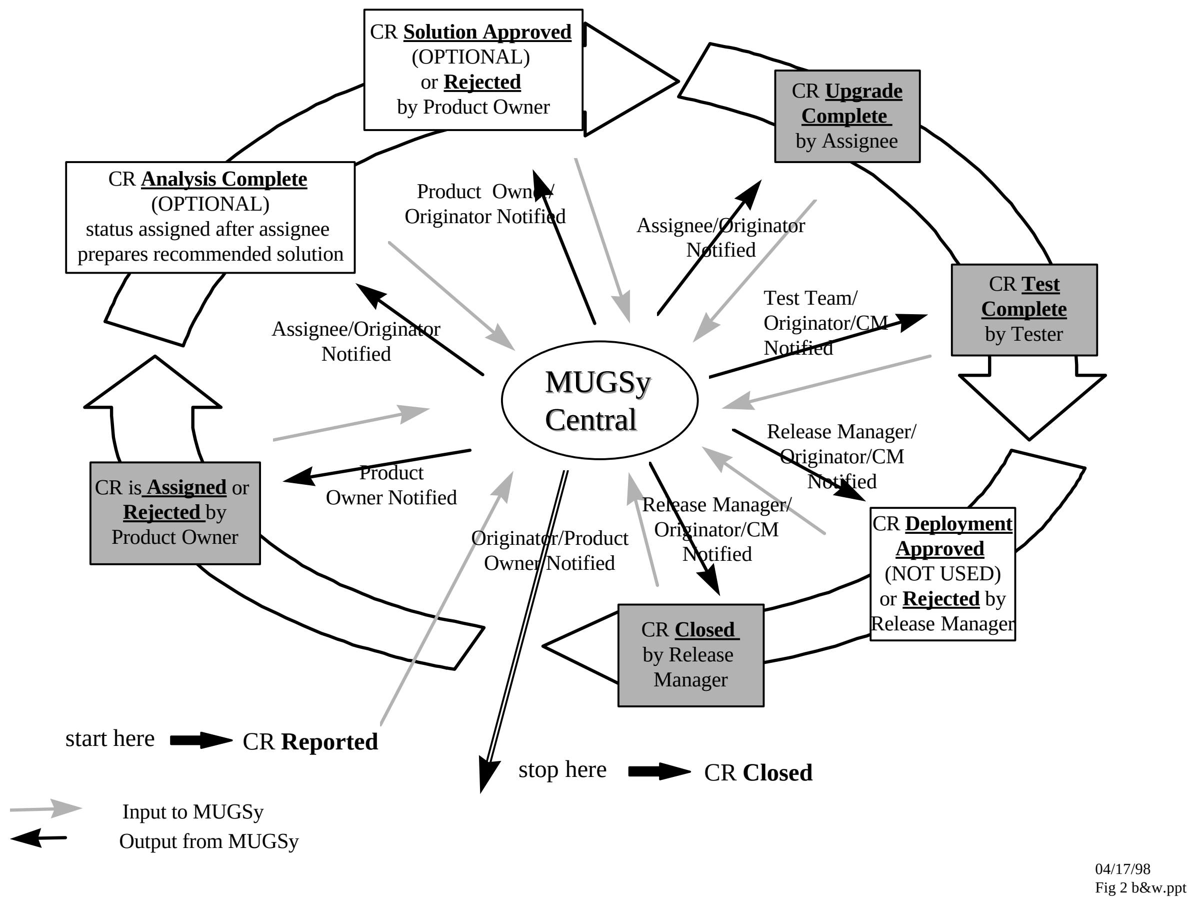
CR Solution Approved (OPTIONAL) or Rejected by Product Owner
CR Upgrade Complete by Assignee
CR Analysis Complete (OPTIONAL) status assigned after assignee prepares recommended solution
Product Owner/ Originator Notified
Assignee/Originator Notified
CR Test Complete by Tester
Test Team/ Originator/CM Notified
Assignee/Originator Notified
MUGSy Central
Release Manager/ Originator/CM Notified
CR is Assigned or Rejected by Product Owner
Product Owner Notified
Release Manager/ Originator/CM Notified
CR Deployment Approved (NOT USED) or Rejected by Release Manager
Originator/Product Owner Notified
CR Closed by Release Manager
start here
CR Reported
stop here
CR Closed
Input to MUGSy
Output from MUGSy
04/17/98
Fig 2 b&w.ppt

Figure 2 CCS Change Request (CR) Process

7. MUGSy EVOLUTION

The initial capability of the MUGSy system was driven by the need to support the CCS development. This capability was labeled the "CCS tracking system" and consisted of the capability to create a CR, update the CR once it was created, and to view the CRs once they were in the system. The same three capabilities (Submit, Update, View) were also provided for the maintenance of user lists (Contact Table), as well as products (Products table) that the CRs were submitted against. MUGSy Release 1 also provided an on-line user’s guide and an primitive reporting capability. The primitive reporting capability displayed the results of a Query-by-example based on search criteria specified using the Find/Update or View Screens.

MUGSy Release 2 was primarily driven by user requests. After the initial CCS Tracking system was released, a need to generate predefined reports was identified. This requirement was satisfied by producing PL/SQL queries that generated the reports in formats that users requested. At the same time, the need to manage CCS and HST Servicing Mission (SM) requirements and track CCS and Servicing Mission test activities was also identified. These needs were satisfied by utilizing the COTS product for creating forms and screens in an HTML format. Based on a request to ingest the CRs for legacy HST software, an HST legacy Change Request Management capability was also implemented.

The most recent evolution of MUGSy, Release 3, responded to the requirement to manage the change process for the HST project database. Command procedures and graphical displays were identified as the first candidates. This requirement was implemented in a very similar manner as the CCS tracking system, but also included an initial integration with the CCS Software Configuration Management product. Another feature implemented during this phase was the addition of a “tester field" that allows a tester to be identified early in the change process and notified throughout the process of any state changes.

Planned capabilities include integrating an Internet based report writer that will allow users to create customized or ad hoc reports, implementing an operations change and tracking system, and integrating the CCS Software Change Management product with the CCS tracking product. This concept of managing the changes to software, commands, and graphical displays with the same system is simple, yet very cost-effective, and not implemented by other GSFC missions.

8. USER REACTIONS

Users have been very receptive to the MUGSy capabilities. Results of a user's poll have been summarized in this section.

The availability of CR information through the Internet appears to be the number one benefit that users agree on. The information in the CRs is critical for some users to perform their job. The fact that this information is available on any platform, from any location that can access the Internet, at any time of the day is a huge selling point. Comments such as "MUGSy puts data at our fingertips" and the "...overall convenience MUGSy provides" indicate that at a basic level, MUGSy is satisfying user needs.

Other features that users have been pleased with are the little things like changing the properties of a text entry field so that it wraps automatically, ensuring that data displayed on a screen will also print out in a readable format, or providing pop-up lists of possible values rather than requiring the user to type in the data. Also, quick responses to upgrade requests go a long way towards encouraging users to use the capabilities of the system.

The area identified most as needing improvement is reporting. Given the need expressed by users to access data, and the limited nature of the reporting tools supplied with the COTS tools used to implement MUGSY, it isn't surprising that the reporting mechanism is cited as the feature that needs the most improvement. Users want (need) their data when and how they want it. The more flexible the reporting system is, the happier users will be with the system.

9. LESSONS LEARNED

This section is written from the developer's point of view. The lessons we have learned fall into 5 categories:

- Development has been a combination of Rapid Application Development/enhancement
- Documenting/organizing COTS customization has been challenging
- Decision to utilize COTS product and implement Internet-based interface contributed significantly to the MUGSy success
- COTS products aren't the be-all and end-all
- Changing the way people do business is extremely challenging

The development cycle has been to complete a RAD/prototype then enhance the prototype depending on user feedback. One week's capability enhancements have sometimes been the previous week's user requests. Getting users to define their requirements was much easier after they had a prototype to use. Developers assumed that the users knew what they wanted. But we found that what they really wanted was in many cases not what they originally requested. We have also found that as we add more capabilities and create inter-related functions, users don't have an overall picture of our system. We need to review requests for their impact to the whole system, not just perform the requested changes without question.

One area that has proved extremely challenging has been that of documenting the COTS product customization. There are two main reasons for the difficulty. One is that there is more work waiting to be done, with users waiting (not always patiently) for their capabilities to be added. So time isn't taken to fully document what was done. The MUGSy CR form has been utilized to document the solution and collect feedback. The second difficulty is that while there is plenty of literature and tools available on how to design and document a new system, not much guidance is available for documenting customization to COTS and representing the interactions between COTS products or between COTS and custom code. The lack of guidance or known procedures as well as how to organize information has hindered us in documenting what we do and in training new personnel. Our current practice is to document the steps or procedures involved in day-to-day operations and development on a local network drive in the hope that some organization will present itself for the documentation once it is created. This continues to be an ongoing effort.

Several decisions that were made early on in the project contributed to the short startup time realized with MUGSy. The first came from the goal of using COTS products where possible to minimize the cost of developing custom solutions. The SCOPUS product was chosen to capitalize on the capabilities of the tool and reduce the development time that would have been required for a custom system. The second resulted from the requirement that the system be available on a wide variety of platforms at various off-site locations. This requirement naturally lends itself to an Internet implementation. The SCOPUS tool supported both a client/server and an Internet implementation, but it was decided to implement strictly an Internet CCS tracking system due to the difficulty of supporting different platforms at different locations.

One drawback with a COTS implementation is that if a desired feature isn't supported, it must be either custom developed or gone without. MUGSy's immediate problem became the difficulty of reporting on the CR data that was in the system. As users became familiar with the MUGSy system, they began to ask for more and more capabilities. They wanted to see the data presented in different ways. MUGSy data was initially presented by generating lists of data based on simple criteria. As our users grew more sophisticated in their use of MUGSy, the capability enhancement requested soon took on the appearance of a custom development shop. The core of MUGSy is the Oracle database management system that the SCOPUS tool is built on. While database systems have been around for years, presenting the information in them on the Internet is a relatively new phenomenon . The problem was initially satisfied by a series of custom reports written in Structured Query Language (SQL) run on an hourly basis that generated HTML web pages. We are currently performing a product review to transition this capability to a COTS product.

From a user’s perspective, changing the way people do business is very challenging. MUGSy is essentially a paperless recording/notification system. The mental transition from seeing a piece of paper go from person to person, to having no paper at all and having automatic e-mail notification when action is required has been difficult for some. There is great reluctance on the part of some users to create a CR when a change is desired. The preferred mode is to wander into the developers office and discuss the change, or make a request over the phone rather than use the system to initiate the change. While this is fine for communicating a need, it doesn’t document the work so it can be better understood, compared with other requests and management priorities assessed.

10. SUMMARY

The success of the MUGSy system is evidenced by the use it has received and the evolution it has experienced. In 2 years, over 2050 CRs have been written, and more importantly, 1500 have been successfully closed. Over 160 users are submitting changes to the system directly, instead of feeding them to administrative support personnel to enter. The time between problem identification to notification of the individual that can act on it has been reduced from weeks to hours. The average time to fully close a CR is 8 weeks, a significant reduction over previous averages. The MUGSy development team uses MUGSy to manage CRs and plan future releases. No additional tool is used. The CCS development team uses MUSGy in a similar manner. The traditional weekly CCB with 3 inches of paper distributed to at least 20 people has been successfully replaced by MUGSy.